IEEE PerCom 2026
Pisa, Italy | March 16 - 20

The 24th IEEE International Conference on Pervasive Computing and Communications

PSMark : A Distributed IoT Benchmark for Publish/Subscribe Under Domain-Based Workloads

Christian Badolato[1], Nathan Samson[1], Houssam Hajj Hassan[2,3], Chih-Kai Huang[2,4], Georgios Bouloukakis[5], Primal Pappachan[6], Roberto Yus[1]

Travel partially funded by

[1] University of Maryland, Baltimore County
[2] Télécom SudParis
[3] Orange Innovation
[4] Télécom Paris
[5] University of Patras
[6] Portland State University

The Publish/Subscribe Paradigm

A communication paradigm for decoupled, event-driven messaging

The Publish/Subscribe Paradigm

A communication paradigm for decoupled, event-driven messaging

- Subscribers (consumers) “subscribe” for events from Publishers (producers)
- Publishers “publish” events to data channels without knowing who is subscribed

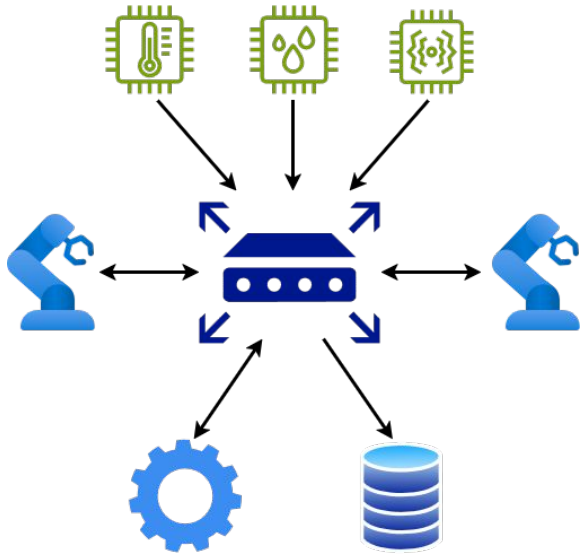
The Publish/Subscribe Paradigm

A communication paradigm for decoupled, event-driven messaging

- Subscribers (consumers) “subscribe” for events from Publishers (producers)
- Publishers “publish” events to data channels without knowing who is subscribed

Broker-based

Routes via a centralized broker



MQTT, AMQP

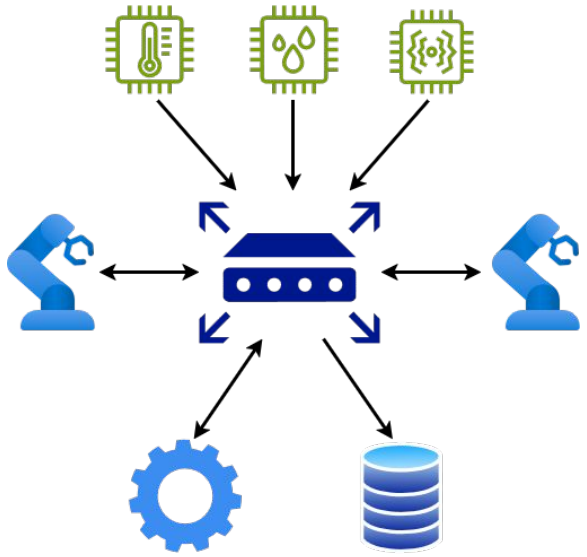
The Publish/Subscribe Paradigm

A communication paradigm for decoupled, event-driven messaging

- Subscribers (consumers) “subscribe” for events from Publishers (producers)
- Publishers “publish” events to data channels without knowing who is subscribed

Broker-based

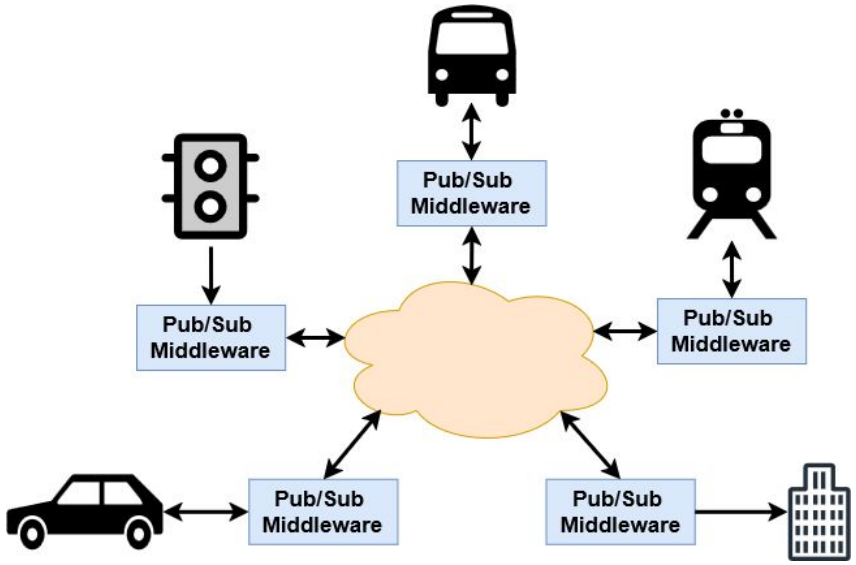
Routes via a centralized broker



MQTT, AMQP

Brokerless

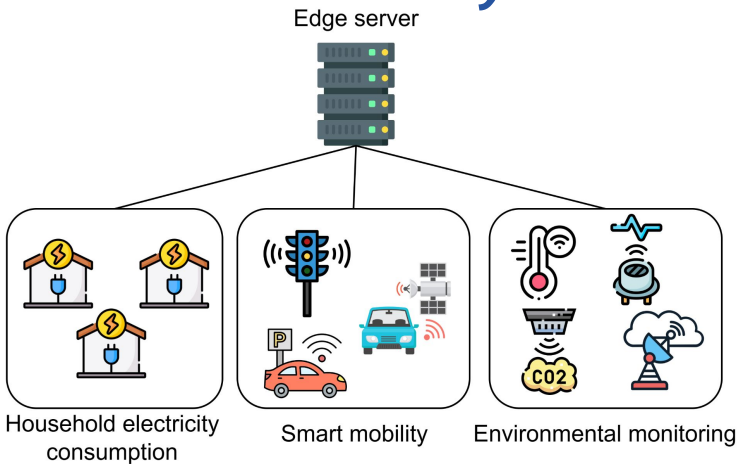
Middleware manages event routing



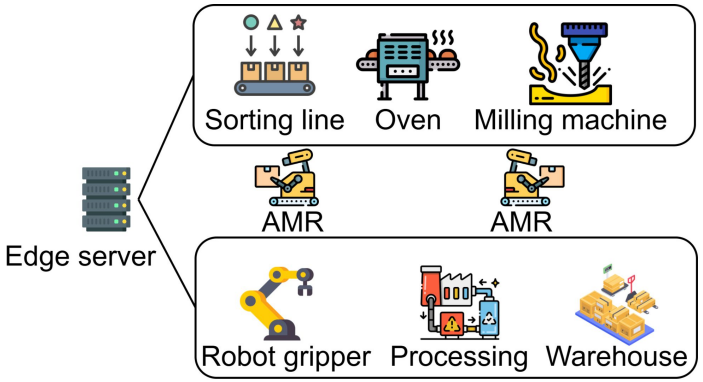
Data Distribution Service (DDS)

Publish/Subscribe in IoT

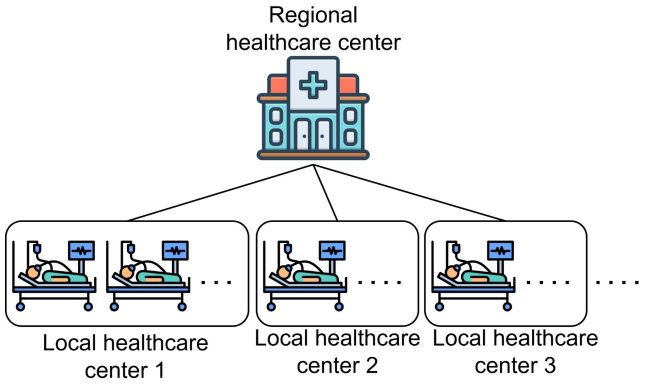
Smart City



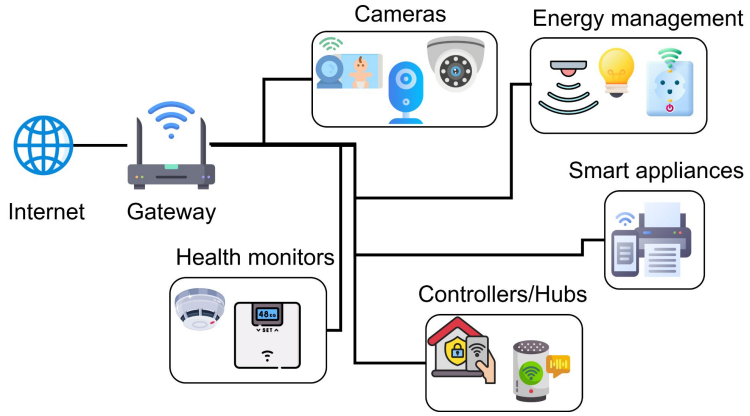
Smart Factory



Smart Healthcare



Smart Homes



Publish/Subscribe in IoT

Pub/sub drives significant amounts of critical infrastructure

Publish/Subscribe in IoT

Pub/sub drives significant amounts of critical infrastructure

Safety Network Management

Hytera Mobilfunk GmbH deploys a country wide public safety network with HiveMQ.

"We are using HiveMQ in a country wide public safety network with 2400 publishers providing up to 2500 messages/ second to central network management applications in near real time..."

Gerald Nolte, CTO at Hytera Mobilfunk

[1]HiveMQ, "Hytera Delivers Secure, Reliable Information for Critical Communications," 2020. https://www.hivemq.com/case-studies/hytera/

Publish/Subscribe in IoT

Pub/sub drives significant amounts of critical infrastructure

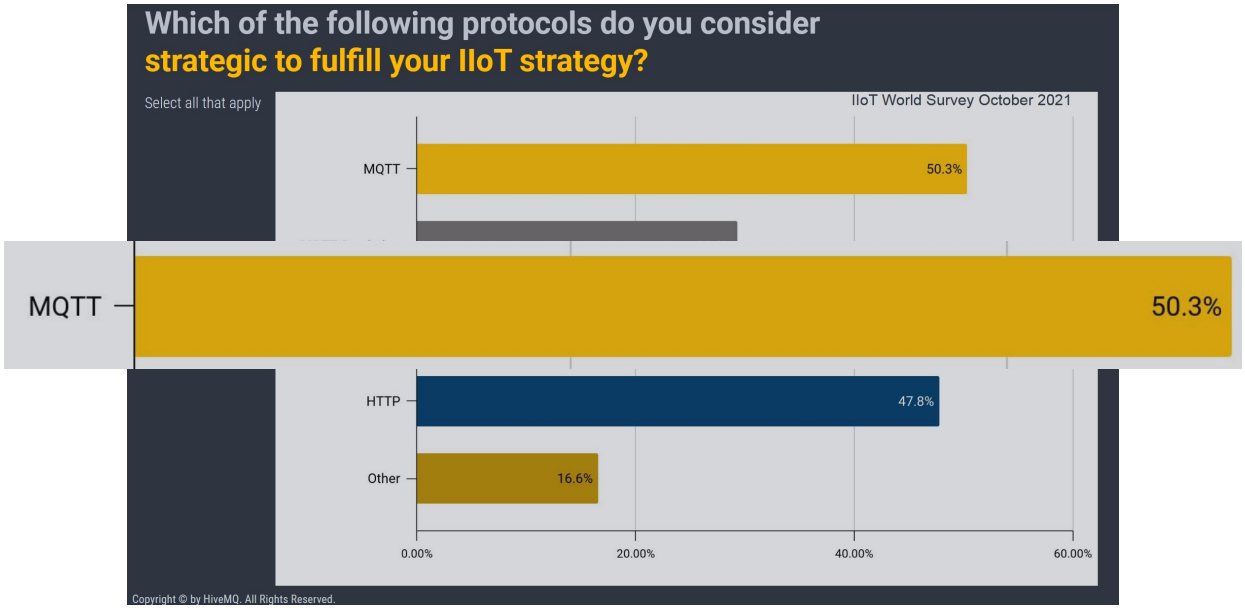
Safety Network Management

Hytera Mobilfunk GmbH deploys a country wide public safety network with HiveMQ.

"We are using HiveMQ in a country wide public safety network with 2400 publishers providing up to 2500 messages/ second to central network management applications in near real time..."

Gerald Nolte, CTO at Hytera Mobilfunk

Industrial Internet of Things



[1]HiveMQ, "Hytera Delivers Secure, Reliable Information for Critical Communications," 2020. https://www.hivemq.com/case-studies/hytera/
[2]IIoT World, HiveMQ, "Trends and Technology Adoption," 2019. https://www.iiot-world.com/industrial-iot/connected-industry/survey-results-building-iiot-systems-trends-and-technology-adoption/

Publish/Subscribe in IoT

Power Grids

RTI Advances Management of Smart Grid IoT Devices with $1M of Department of Energy Funding

RTI Collaborates with Duke Energy and SEPA to Leverage the Data Distribution Service™ (DDS) and Open Field Message Bus (OpenFMB™) Standards

[1]Real-Time Innovations, "RTI Advances Management of Smart Grid IoT Devices with $1M of Department of Energy Funding" 2017. https://www.rti.com/news/rti-advances-management-of-smart-grid-iot-devices-with-1m-of-department-of-energy-funding

Publish/Subscribe in IoT

Power Grids

RTI Advances Management of Smart Grid IoT Devices with $1M of Department of Energy Funding

RTI Collaborates with Duke Energy and SEPA to Leverage the Data Distribution Service™ (DDS) and Open Field Message Bus (OpenFMB™) Standards

Transportation

ProRail Deploys Vortex OpenSplice for Dutch Railway Network.

Vortex OpenSplice provides ProRail with a reliable, real-time and fault-tolerant data-sharing platform to manage critical information within the railway system. With more than six thousand trains and 1.2 million passengers travelling on it on a daily basis, the Netherlands railway system is one of the busiest in Europe.

Vortex OpenSplice is the leading commercial and Open Source implementation of the Object Management Group™'s (OMG™) Data Distribution Service (DDS) for Real-Time Systems standard.

[1]Real-Time Innovations, "RTI Advances Management of Smart Grid IoT Devices with $1M of Department of Energy Funding" 2017. https://www.rti.com/news/rti-advances-management-of-smart-grid-iot-devices-with-1m-of-department-of-energy-funding
[2]Adlink, "ProRail Deploys Vortext OpenSplice for Dutch Railway Network," 2021. https://www.adlinktech.com/en/ProRail.aspx

Publish/Subscribe in IoT

Power Grids

RTI Advances Management of Smart Grid IoT Devices with $1M of Department of Energy Funding

RTI Collaborates with Duke Energy and SEPA to Leverage the Data Distribution Service™ (DDS) and Open Field Message Bus (OpenFMB™) Standards

Transportation

ProRail Deploys Vortex OpenSplice for Dutch Railway Network.

Vortex OpenSplice provides ProRail with a reliable, real-time and fault-tolerant data-sharing platform to manage critical information

1.2 million passengers travelling on it on a daily basis

Vortex OpenSplice is the leading commercial and Open Source implementation of the Object Management Group™'s (OMG™) Data Distribution Service (DDS) for Real-Time Systems standard.

[1]Real-Time Innovations, "RTI Advances Management of Smart Grid IoT Devices with $1M of Department of Energy Funding" 2017. https://www.rti.com/news/rti-advances-management-of-smart-grid-iot-devices-with-1m-of-department-of-energy-funding
[2]Adlink, "ProRail Deploys Vortext OpenSplice for Dutch Railway Network," 2021. https://www.adlinktech.com/en/ProRail.aspx

Publish/Subscribe in IoT

Power Grids

RTI Advances Management of Smart Grid IoT Devices with $1M of Department of Energy Funding

RTI Collaborates with Duke Energy and SEPA to Leverage the Data Distribution Service™ (DDS) and Open Field Message Bus (OpenFMB™) Standards

Transportation

ProRail Deploys Vortex OpenSplice for Dutch Railway Network.

Vortex OpenSplice provides ProRail with a reliable, real-time and fault-tolerant data-sharing platform to manage critical information

1.2 million passengers travelling on it on a daily basis

Vortex OpenSplice is the leading commercial and Open Source implementation of the Object Management Group™'s (OMG™) Data Distribution Service (DDS) for Real-Time Systems standard.

Failure of an actively deployed pub/sub system could be catastrophic!
Evaluation under real-world conditions must occur prior to deployment

[1]Real-Time Innovations, "RTI Advances Management of Smart Grid IoT Devices with $1M of Department of Energy Funding" 2017. https://www.rti.com/news/rti-advances-management-of-smart-grid-iot-devices-with-1m-of-department-of-energy-funding
[2]Adlink, "ProRail Deploys Vortext OpenSplice for Dutch Railway Network," 2021. https://www.adlinktech.com/en/ProRail.aspx

Do We Need a New IoT-focused Pub/Sub Benchmark?

Do We Need a New IoT-focused Pub/Sub Benchmark?

Pub/sub systems in IoT are not one-size-fits-all

- Vastly different device types, quantities, data volumes, and processing power

Do We Need a New IoT-focused Pub/Sub Benchmark?

Pub/sub systems in IoT are not one-size-fits-all

- Vastly different device types, quantities, data volumes, and processing power

Limitations of existing pub/sub-focused benchmarks include:

Feature	[36]	[37]	[21]	[25]	[26]	[27]	[38]	[22]	[30]	[39]	[28]	[29]	PSMark
Pub/Sub Focused	✓	✓	✓				✓	✓	✓		✓	✓	✓
Multiple Protocol Support				✓	✓	✓			✓		✓	✓	✓
Brokerless Protocol Support		✓		Partial*	Partial*	Partial*	✓		✓				✓
Measures Distributed Topologies							✓	✓		✓		✓	✓
Built-In Workloads						✓	✓			✓		Partial†	✓
IoT Focused										✓		✓	✓
Real-World Device Behaviors										✓		Partial‡	✓
Extensibility				✓	✓	✓					✓	Partial§	✓

* Requires plugin development - † Workload is retired - ‡ Does not model connectivity or variable message attributes - § Cannot extend metrics

Do We Need a New IoT-focused Pub/Sub Benchmark?

Pub/sub systems in IoT are not one-size-fits-all

- Vastly different device types, quantities, data volumes, and processing power

Limitations of existing pub/sub-focused benchmarks include:

- Only support a single protocol

Feature	[36]	[37]	[21]	[25]	[26]	[27]	[38]	[22]	[30]	[39]	[28]	[29]	PSMark
Pub/Sub Focused	✓	✓	✓				✓	✓	✓		✓	✓	✓
Multiple Protocol Support				✓	✓	✓			✓		✓	✓	✓
Brokerless Protocol Support		✓		Partial*	Partial*	Partial*	✓		✓				✓
Measures Distributed Topologies							✓	✓		✓		✓	✓
Built-In Workloads						✓	✓			✓		Partial†	✓
IoT Focused										✓		✓	✓
Real-World Device Behaviors										✓		Partial‡	✓
Extensibility				✓	✓	✓					✓	Partial§	✓

* Requires plugin development - † Workload is retired - ‡ Does not model connectivity or variable message attributes - § Cannot extend metrics

Do We Need a New IoT-focused Pub/Sub Benchmark?

Pub/sub systems in IoT are not one-size-fits-all

- Vastly different device types, quantities, data volumes, and processing power

Limitations of existing pub/sub-focused benchmarks include:

- Only support a single protocol
- Limited representation for large numbers of distributed devices

Feature	[36]	[37]	[21]	[25]	[26]	[27]	[38]	[22]	[30]	[39]	[28]	[29]	PSMark
Pub/Sub Focused	✓	✓	✓				✓	✓	✓		✓	✓	✓
Multiple Protocol Support				✓	✓	✓			✓		✓	✓	✓
Brokerless Protocol Support		✓		Partial*	Partial*	Partial*	✓		✓				✓
Measures Distributed Topologies							✓	✓		✓		✓	✓
Built-In Workloads						✓	✓			✓		Partial†	✓
IoT Focused										✓		✓	✓
Real-World Device Behaviors										✓		Partial‡	✓
Extensibility				✓	✓	✓					✓	Partial§	✓

* Requires plugin development - † Workload is retired - ‡ Does not model connectivity or variable message attributes - § Cannot extend metrics

Do We Need a New IoT-focused Pub/Sub Benchmark?

Pub/sub systems in IoT are not one-size-fits-all

- Vastly different device types, quantities, data volumes, and processing power

Limitations of existing pub/sub-focused benchmarks include:

- Only support a single protocol
- Limited representation for large numbers of distributed devices
- Static configurations that cannot express IoT device heterogeneity

Feature	[36]	[37]	[21]	[25]	[26]	[27]	[38]	[22]	[30]	[39]	[28]	[29]	PSMark
Pub/Sub Focused	✓	✓	✓				✓	✓	✓		✓	✓	✓
Multiple Protocol Support				✓	✓	✓			✓		✓	✓	✓
Brokerless Protocol Support		✓		Partial*	Partial*	Partial*	✓		✓				✓
Measures Distributed Topologies							✓	✓		✓		✓	✓
Built-In Workloads						✓	✓			✓		Partial†	✓
IoT Focused										✓		✓	✓
Real-World Device Behaviors										✓		Partial‡	✓
Extensibility				✓	✓	✓					✓	Partial§	✓

* Requires plugin development - † Workload is retired - ‡ Does not model connectivity or variable message attributes - § Cannot extend metrics

Do We Need a New IoT-focused Pub/Sub Benchmark?

Pub/sub systems in IoT are not one-size-fits-all

- Vastly different device types, quantities, data volumes, and processing power

Limitations of existing pub/sub-focused benchmarks include:

- Only support a single protocol
- Limited representation for large numbers of distributed devices
- Static configurations that cannot express IoT device heterogeneity
- Focus on stress testing rather than real-world device behaviors

Feature	[36]	[37]	[21]	[25]	[26]	[27]	[38]	[22]	[30]	[39]	[28]	[29]	PSMark
Pub/Sub Focused	✓	✓	✓				✓	✓	✓		✓	✓	✓
Multiple Protocol Support				✓	✓	✓			✓		✓	✓	✓
Brokerless Protocol Support		✓		Partial*	Partial*	Partial*	✓		✓				✓
Measures Distributed Topologies							✓	✓		✓		✓	✓
Built-In Workloads						✓	✓			✓		Partial†	✓
IoT Focused										✓		✓	✓
Real-World Device Behaviors										✓		Partial‡	✓
Extensibility				✓	✓	✓					✓	Partial§	✓

* Requires plugin development - † Workload is retired - ‡ Does not model connectivity or variable message attributes - § Cannot extend metrics

PSMark provides a distributed, IoT pub/sub benchmark incorporating:

PSMark provides a distributed, IoT pub/sub benchmark incorporating:

i. Multi-protocol support

PSMark provides a distributed, IoT pub/sub benchmark incorporating:

i. Multi-protocol support
ii. Distributed topology evaluation

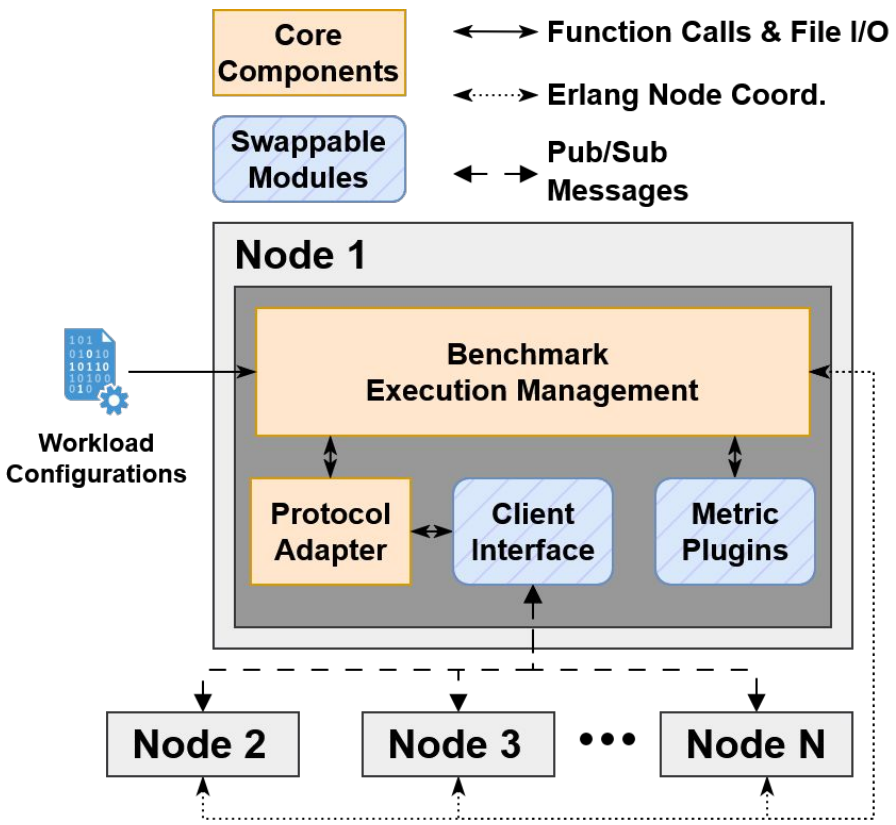
PSMark provides a distributed, IoT pub/sub benchmark incorporating:

i. Multi-protocol support
ii. Distributed topology evaluation
iii. Explicit modeling of device heterogeneity and connection stability

PSMark provides a distributed, IoT pub/sub benchmark incorporating:

i. Multi-protocol support
ii. Distributed topology evaluation
iii. Explicit modeling of device heterogeneity and connection stability
iv. An open set of IoT-focused workloads derived from real-world datasets

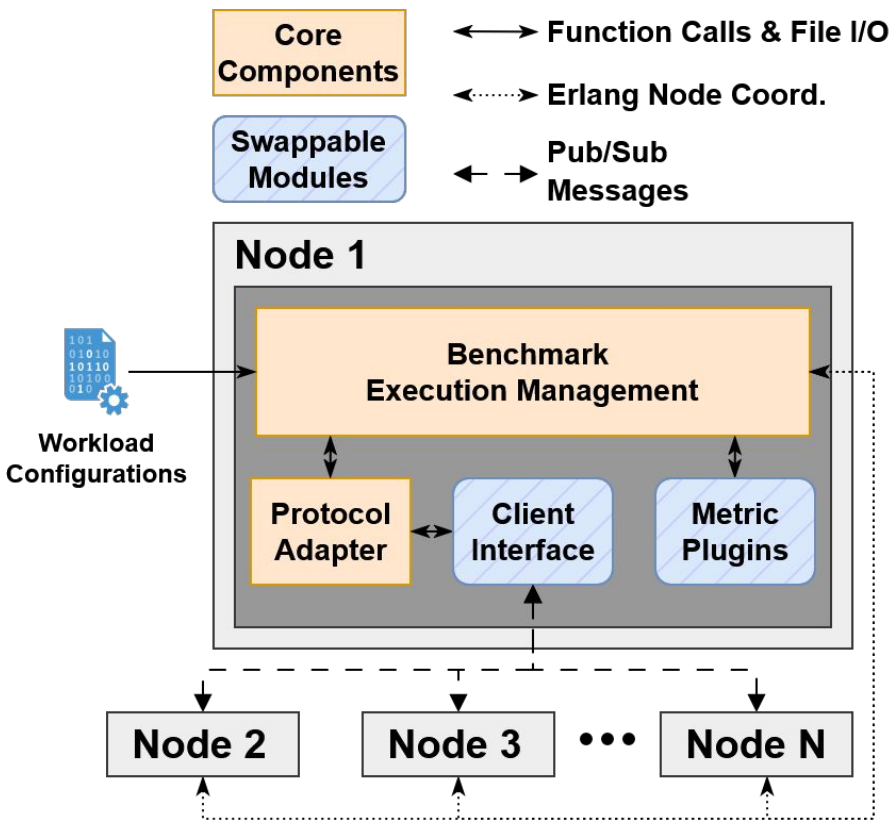
PSMark Architecture



Coordinates across distributed nodes

- Analogous to IoT deployment edge servers

PSMark Architecture



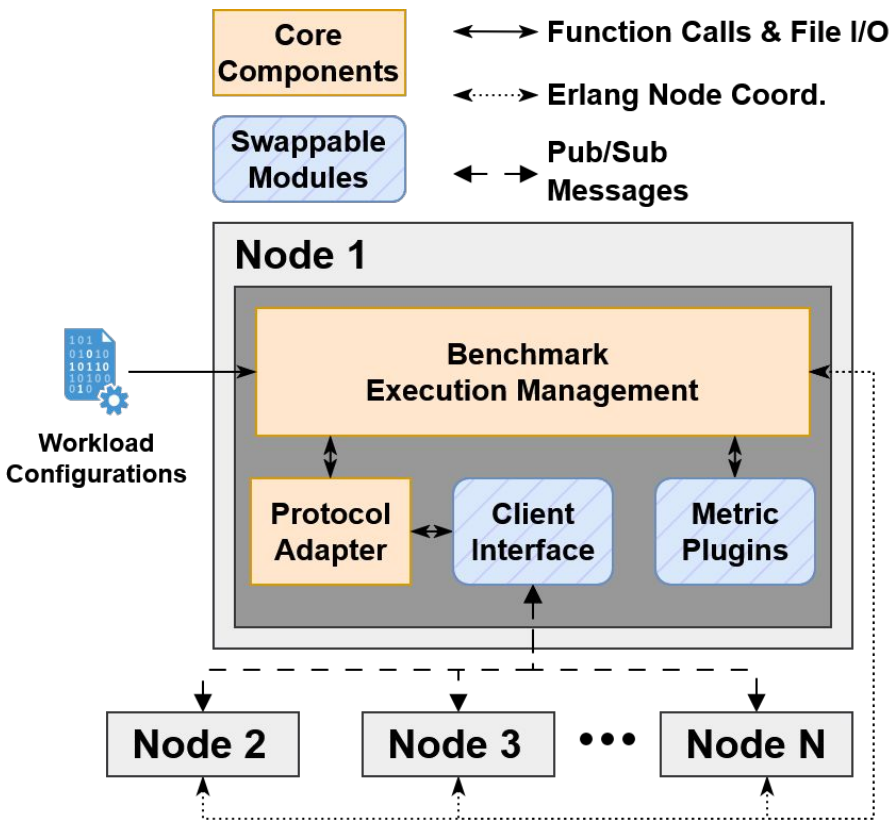
Coordinates across distributed nodes

- Analogous to IoT deployment edge servers

Nodes independently manage:

- A publisher pool, which generates workload traffic
- A subscriber pool as a data aggregation point
  - e.g., databases, applications, AIoT models

PSMark Architecture



Coordinates across distributed nodes

- Analogous to IoT deployment edge servers

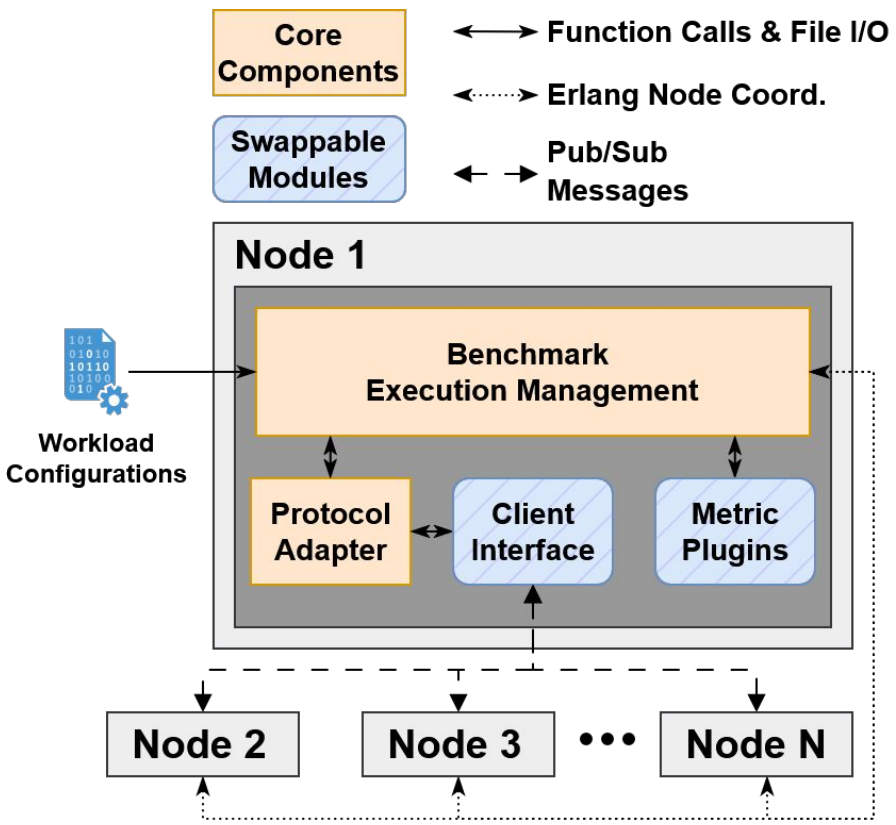
Nodes independently manage:

- A publisher pool, which generates workload traffic
- A subscriber pool as a data aggregation point
  - e.g., databases, applications, AIoT models

Adapter-based, modular architecture

- *Protocol Adapters*: Translate high-level pub/sub operations into protocol-specific commands

PSMark Architecture



Coordinates across distributed nodes

- Analogous to IoT deployment edge servers

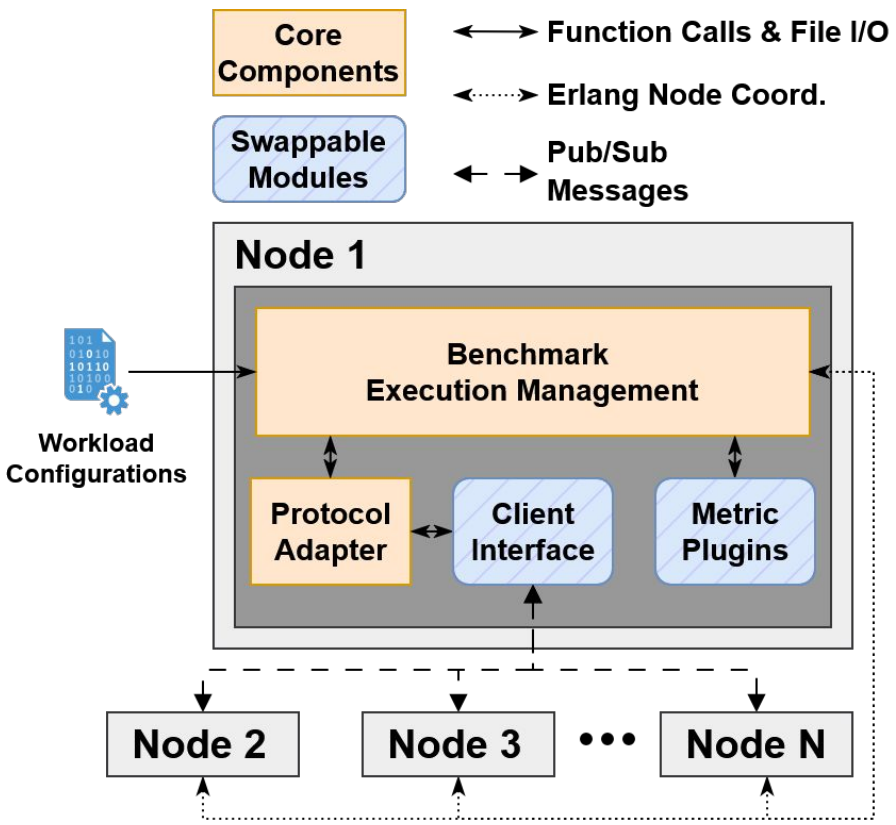
Nodes independently manage:

- A publisher pool, which generates workload traffic
- A subscriber pool as a data aggregation point
  - e.g., databases, applications, AIoT models

Adapter-based, modular architecture

- *Protocol Adapters*: Translate high-level pub/sub operations into protocol-specific commands
- *Client interfaces*: Execute protocol commands against concrete libraries and perform pre-/post-processing

PSMark Architecture



Coordinates across distributed nodes

- Analogous to IoT deployment edge servers

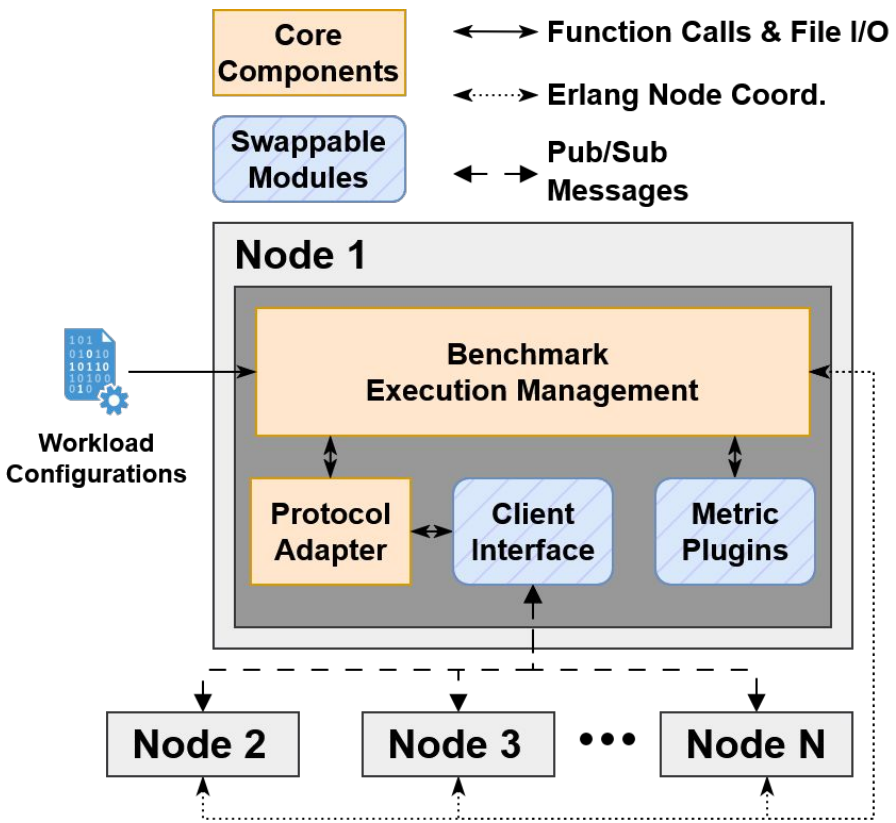
Nodes independently manage:

- A publisher pool, which generates workload traffic
- A subscriber pool as a data aggregation point
  - e.g., databases, applications, AIoT models

Adapter-based, modular architecture

- *Protocol Adapters*: Translate high-level pub/sub operations into protocol-specific commands
- *Client interfaces*: Execute protocol commands against concrete libraries and perform pre-/post-processing
- *Metric Plugins*: Simplify the inclusion of new metrics

PSMark Architecture



Coordinates across distributed nodes

- Analogous to IoT deployment edge servers

Nodes independently manage:

- A publisher pool, which generates workload traffic
- A subscriber pool as a data aggregation point
  - e.g., databases, applications, AIoT models

Adapter-based, modular architecture

- *Protocol Adapters*: Translate high-level pub/sub operations into protocol-specific commands
- *Client interfaces*: Execute protocol commands against concrete libraries and perform pre-/post-processing
- *Metric Plugins*: Simplify the inclusion of new metrics

Built-in MQTT and DDS modules
Built-in latency, throughput, message loss, and CPU/RAM usage metrics

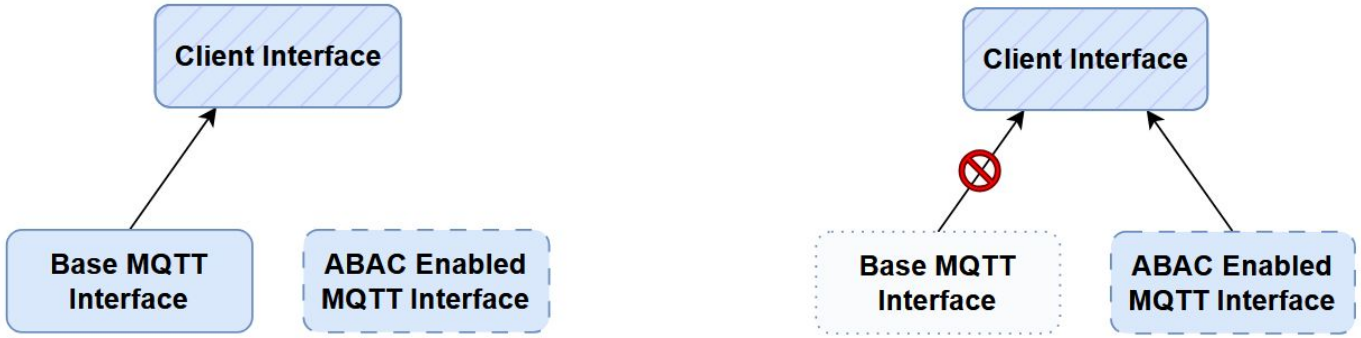
Extensibility to Varied Systems

Swappable modules enable support for application-specific logic

Extensibility to Varied Systems

Swappable modules enable support for application-specific logic

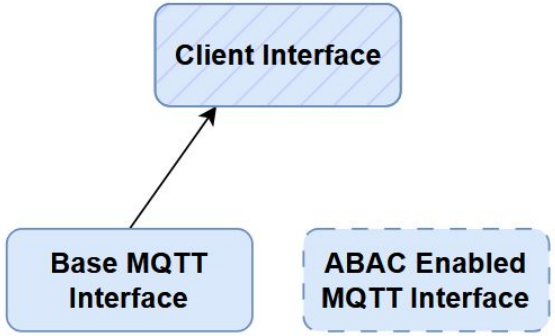
Client Interfaces:

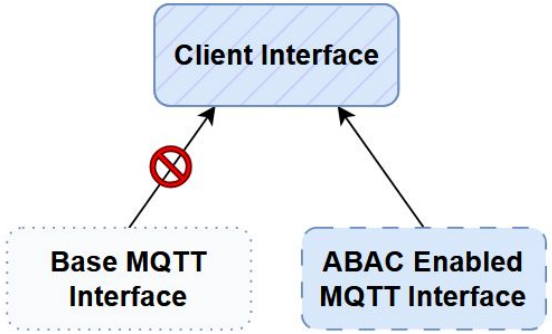


Extensibility to Varied Systems

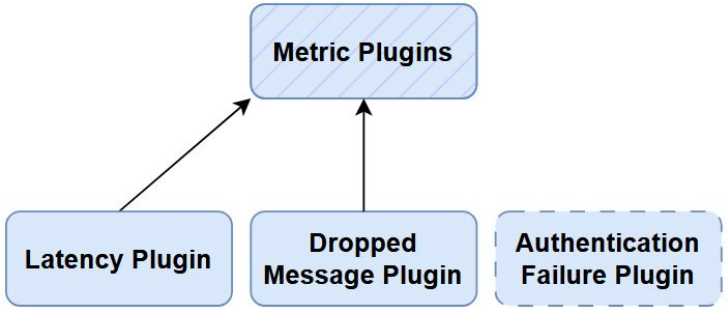
Swappable modules enable support for application-specific logic

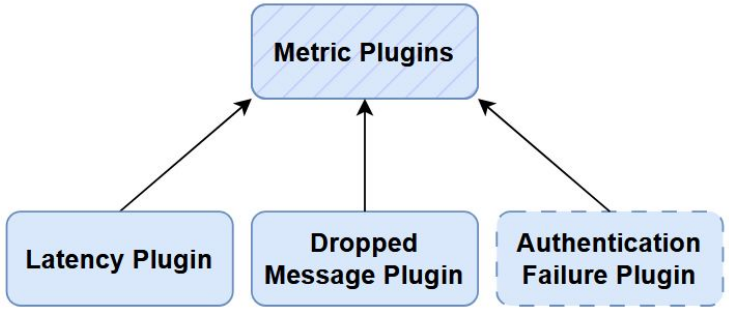
Client Interfaces:



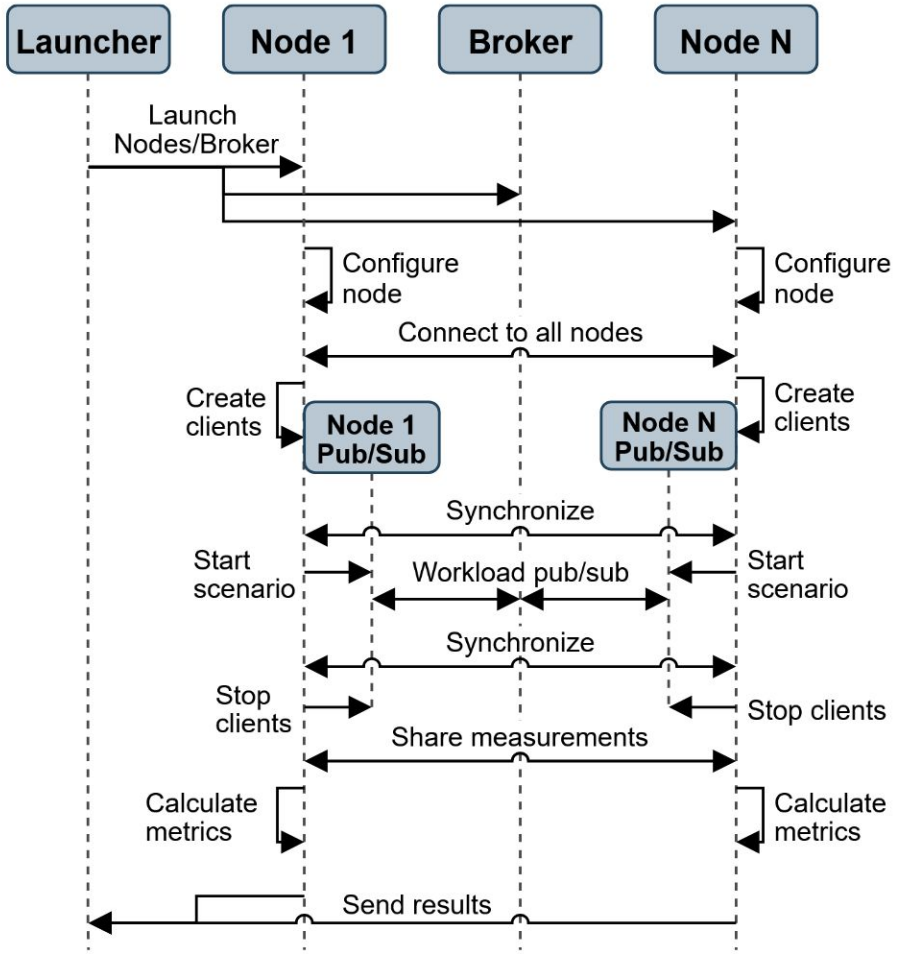


Metric Plugins:

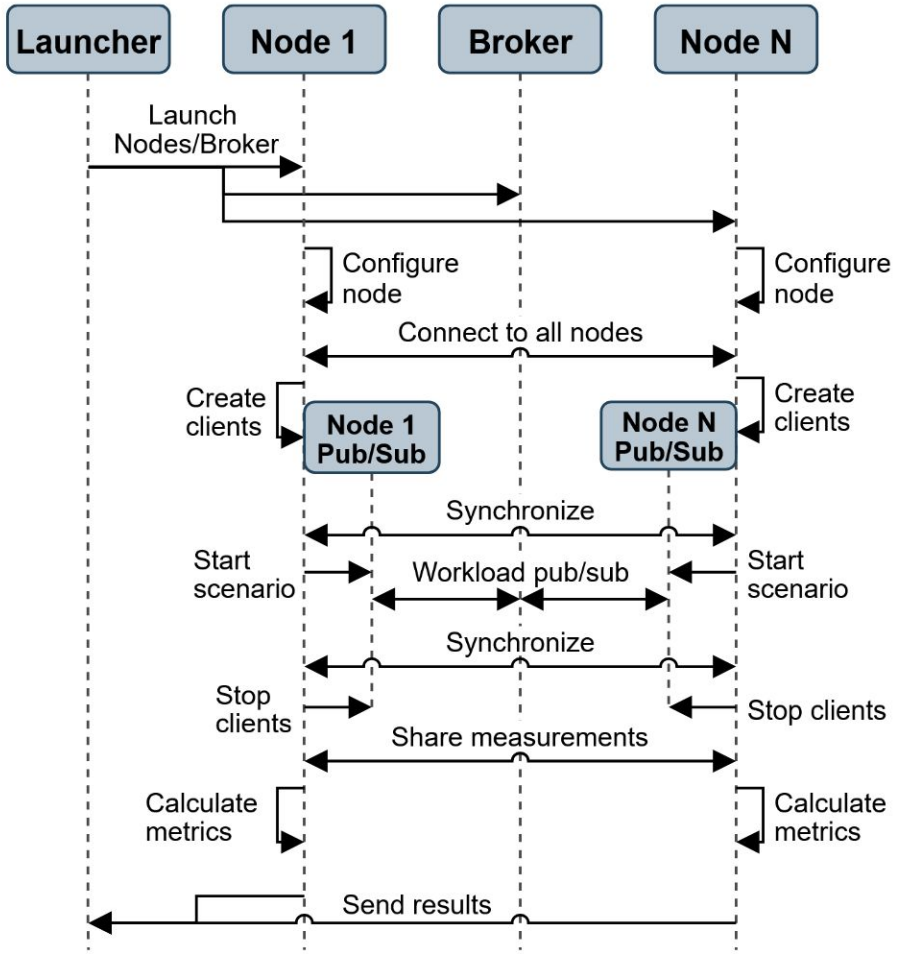




Benchmark Execution Flow



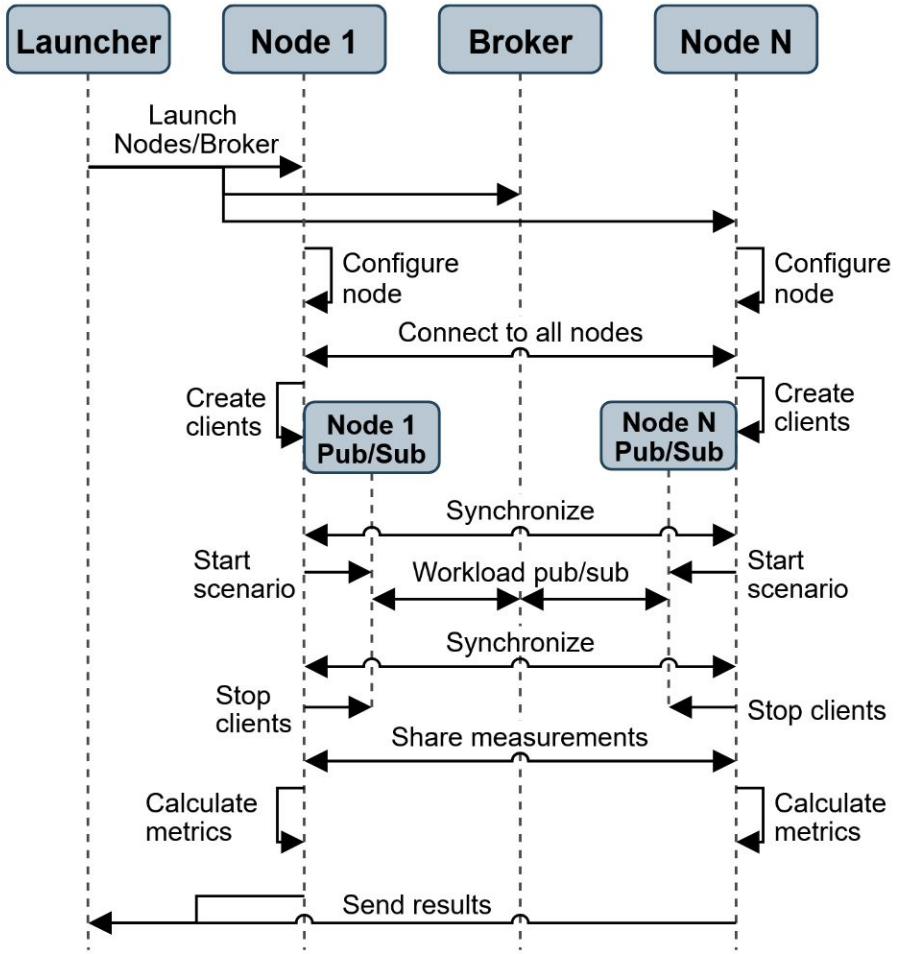
Benchmark Execution Flow



Barrier-synchronized lifecycle

- Nodes must all acknowledge stage completion before any node proceeds to the next stage
- Ensures subscribers are active prior to workload start and and remain subscribed until workload completion

Benchmark Execution Flow



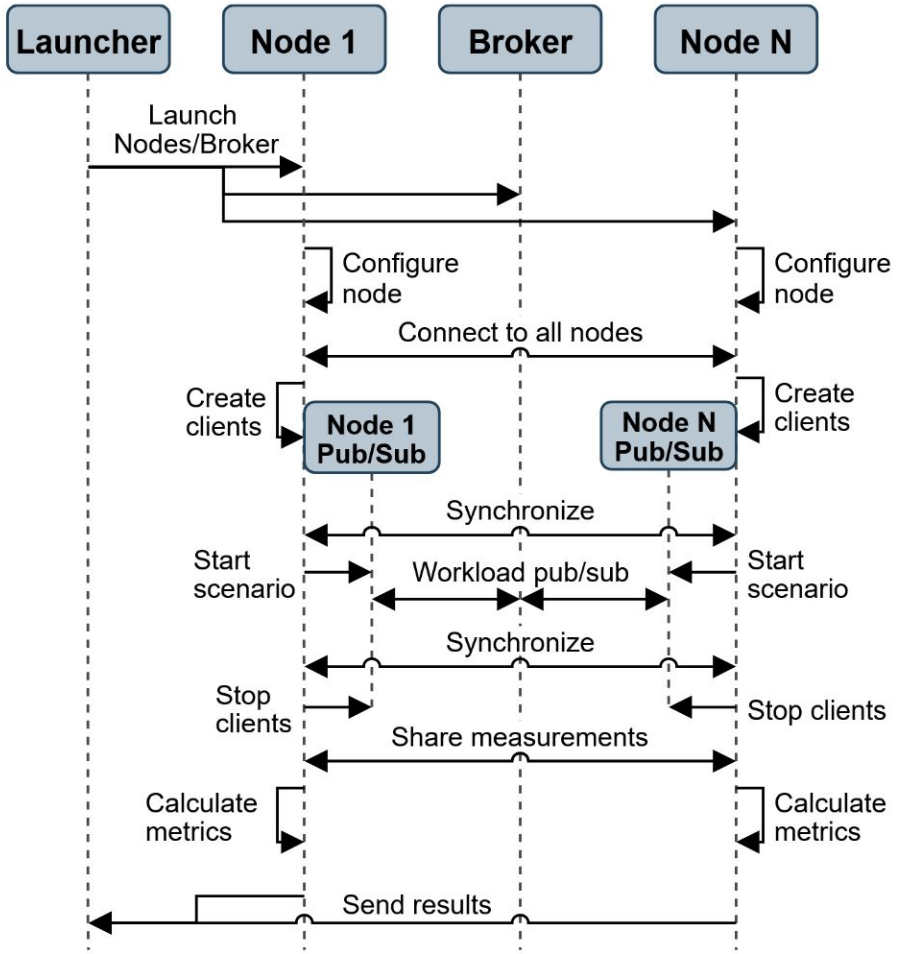
Barrier-synchronized lifecycle

- Nodes must all acknowledge stage completion before any node proceeds to the next stage
- Ensures subscribers are active prior to workload start and and remain subscribed until workload completion

Nodes each report their own metrics:

- Ensures single nodes do not bias global aggregates
- Pairwise, node-to-node metrics are collected to measure network path behaviors

Benchmark Execution Flow



Barrier-synchronized lifecycle

- Nodes must all acknowledge stage completion before any node proceeds to the next stage
- Ensures subscribers are active prior to workload start and and remain subscribed until workload completion

Nodes each report their own metrics:

- Ensures single nodes do not bias global aggregates
- Pairwise, node-to-node metrics are collected to measure network path behaviors

Written with Erlang/OTP

- Provides lightweight multithreading and efficient inter-process communication with minimal overhead
- Separates coordination from pub/sub traffic

Workload Expression

Workloads are composed of reusable, configurable device definitions

- Publication frequency
- Payload size parameters
- **Stability parameters**
  - e.g., Cars leaving/re-entering a smart city

Workload Expression

Workloads are composed of reusable, configurable device definitions

- Publication frequency
- Payload size parameters
- **Stability parameters**
  - e.g., Cars leaving/re-entering a smart city

Example Smart Factory Devices

```
Device: Machine Temperature Sensor
Publication Rate: 1 second
Mean Payload Size: 94B
Payload Size Variance: 5B
Stability: 5% disconnect per 1s
           80% reconnect per 1s
```

```
Device: Robot Lidar
Publication Rate: 100 milliseconds
Mean Payload Size: 1.2MB
Payload Size Variance: 1MB
Stability: 5% disconnect per 1s
           80% reconnect per 1s
```

Workload Expression

Deployments are defined to mimic the IoT system topology

- Edge servers in the deployment
- Type / quantity of devices present on the edge servers

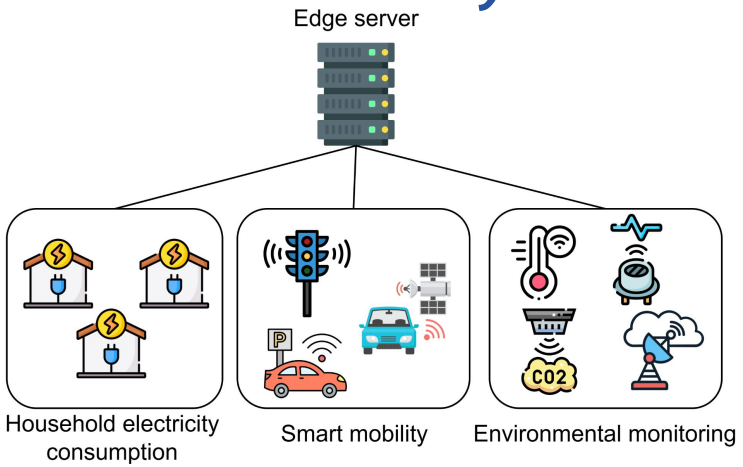
```
Deployment Name: Two Server Smart Factory

Edge Servers:
 -  Floor 1 Server: 10x Machine Temperature Sensors, 3x Robot Lidar
 -  Floor 2 Server: 5x Machine Temperature Sensors, 6x Robot Lidar
```

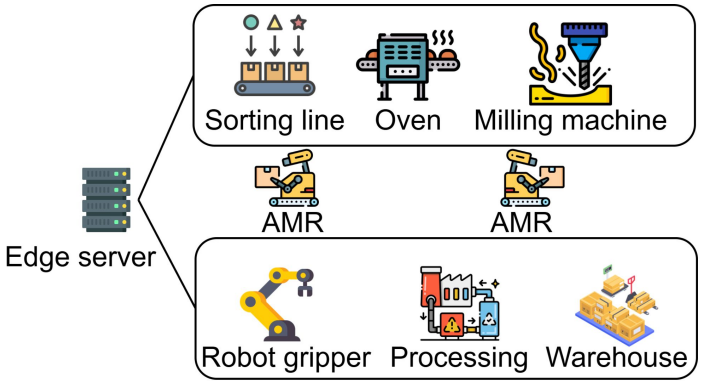
PSMark Built-in Workloads

PSMark provides 12 built-in workloads targeting 4 IoT domains

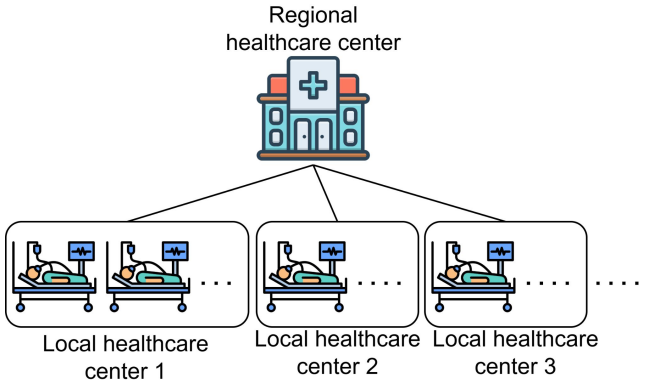
Smart City



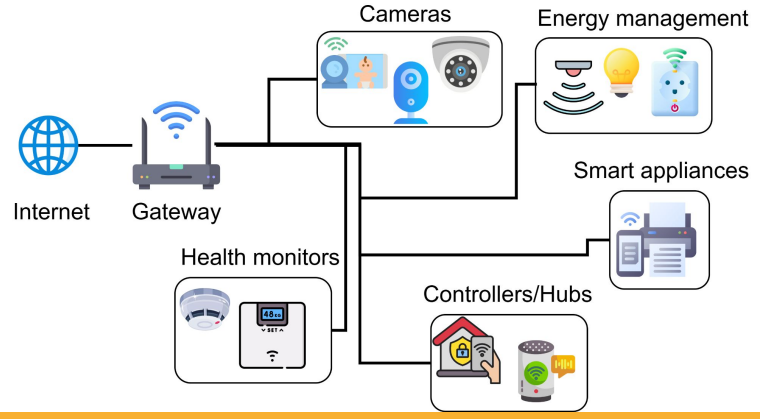
Smart Factory



Smart Healthcare



Smart Homes



PSMark Built-in Workloads

PSMark provides 12 built-in workloads targeting 4 IoT domains

Device attributes were derived from 7 real-world IoT datasets*

*R. Hernangómez et al., "Toward an AI-Enabled Connected Industry: AGV Communication and Sensor Measurement Datasets," 2024. https://ieeexplore.ieee.org/document/10494952
A. Mahajan, "Pune Smart City Dataset," 2022. https://kaggle.com/datasets/akshman/pune-smartcity-test-dataset
A. Sivanathan et al., "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," 2018. https://ieeexplore.ieee.org/document/8440758
S. Agrawal et al, "High frequency smart meter data from two districts in India (Mathura and Bareilly)," 2022. https://doi.org/10.7910/DVN/GOCHJH
Python Developer, "Smart Manufacturing Process Data," 2025. https://www.kaggle.com/datasets/programmer3/smart-manufacturing-process-data
Ziya, "Smart Mobility Traffic Dataset," 2025. https://kaggle.com/datasets/ziya07/smart-mobility-traffic-dataset
V. Kumar, "IoT in Healthcare & Well-being," 2025. https://www.kaggle.com/datasets/dcsavinod/iot-in-healthcare-and-well-being

PSMark Built-in Workloads

PSMark provides 12 built-in workloads targeting 4 IoT domains

Device attributes were derived from 7 real-world IoT datasets*

Three workloads for each domain

- Comprised of 1x (base), 2x, and 10x device density scaling
- Workload-wide stability parameters

Workload	Devices	Msgs/s	Disconnection Rate	Reconnection Rate
PSMark-C	541	109	20% per 5s	50% per 5s
PSMark-C-2x	1,082	218	20% per 5s	50% per 5s
PSMark-C-10x	5,410	1,090	20% per 5s	50% per 5s
PSMark-F	40	590	5% per 1s	80% per 1s
PSMark-F-2x	80	1,180	5% per 1s	80% per 1s
PSMark-F-10x	400	5,900	5% per 1s	80% per 1s
PSMark-HC	24	5	10% per 5s	80% per 5s
PSMark-HC-2x	48	10	10% per 5s	80% per 5s
PSMark-HC-10x	240	50	10% per 5s	80% per 5s
PSMark-HM	20	148	5% per 1s	80% per 1s
PSMark-HM-2x	40	296	5% per 1s	80% per 1s
PSMark-HM-10x	200	1,480	5% per 1s	80% per 1s

*R. Hernangómez et al., "Toward an AI-Enabled Connected Industry: AGV Communication and Sensor Measurement Datasets," 2024. https://ieeexplore.ieee.org/document/10494952
A. Mahajan, "Pune Smart City Dataset," 2022. https://kaggle.com/datasets/akshman/pune-smartcity-test-dataset
A. Sivanathan et al., "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," 2018. https://ieeexplore.ieee.org/document/8440758
S. Agrawal et al, "High frequency smart meter data from two districts in India (Mathura and Bareilly)," 2022. https://doi.org/10.7910/DVN/GOCHJH
Python Developer, "Smart Manufacturing Process Data," 2025. https://www.kaggle.com/datasets/programmer3/smart-manufacturing-process-data
Ziya, "Smart Mobility Traffic Dataset," 2025. https://kaggle.com/datasets/ziya07/smart-mobility-traffic-dataset
V. Kumar, "IoT in Healthcare & Well-being," 2025. https://www.kaggle.com/datasets/dcsavinod/iot-in-healthcare-and-well-being

PSMark Built-in Workloads

PSMark provides 12 built-in workloads targeting 4 IoT domains

Device attributes were derived from 7 real-world IoT datasets*

Three workloads for each domain

- Comprised of 1x (base), 2x, and 10x device density scaling
- Workload-wide stability parameters

Workloads represent device load per edge server

Workload	Devices	Msgs/s	Disconnection Rate	Reconnection Rate
PSMark-C	541	109	20% per 5s	50% per 5s
PSMark-C-2x	1,082	218	20% per 5s	50% per 5s
PSMark-C-10x	5,410	1,090	20% per 5s	50% per 5s
PSMark-F	40	590	5% per 1s	80% per 1s
PSMark-F-2x	80	1,180	5% per 1s	80% per 1s
PSMark-F-10x	400	5,900	5% per 1s	80% per 1s
PSMark-HC	24	5	10% per 5s	80% per 5s
PSMark-HC-2x	48	10	10% per 5s	80% per 5s
PSMark-HC-10x	240	50	10% per 5s	80% per 5s
PSMark-HM	20	148	5% per 1s	80% per 1s
PSMark-HM-2x	40	296	5% per 1s	80% per 1s
PSMark-HM-10x	200	1,480	5% per 1s	80% per 1s

*R. Hernangómez et al., "Toward an AI-Enabled Connected Industry: AGV Communication and Sensor Measurement Datasets," 2024. https://ieeexplore.ieee.org/document/10494952
A. Mahajan, "Pune Smart City Dataset," 2022. https://kaggle.com/datasets/akshman/pune-smartcity-test-dataset
A. Sivanathan et al., "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics," 2018. https://ieeexplore.ieee.org/document/8440758
S. Agrawal et al, "High frequency smart meter data from two districts in India (Mathura and Bareilly)," 2022. https://doi.org/10.7910/DVN/GOCHJH
Python Developer, "Smart Manufacturing Process Data," 2025. https://www.kaggle.com/datasets/programmer3/smart-manufacturing-process-data
Ziya, "Smart Mobility Traffic Dataset," 2025. https://kaggle.com/datasets/ziya07/smart-mobility-traffic-dataset
V. Kumar, "IoT in Healthcare & Well-being," 2025. https://www.kaggle.com/datasets/dcsavinod/iot-in-healthcare-and-well-being

Benchmarking Popular Pub/Sub Systems

We evaluate 6 pub/sub systems using PSMark

Benchmarking Popular Pub/Sub Systems

We evaluate 6 pub/sub systems using PSMark

- Five MQTT systems (broker-based)
  1. EMQX
  2. Mochi-MQTT
  3. Eclipse Mosquitto
  4. NanoMQ
  5. VerneMQ

Benchmarking Popular Pub/Sub Systems

We evaluate 6 pub/sub systems using PSMark

- Five MQTT systems (broker-based)
  1. EMQX
  2. Mochi-MQTT
  3. Eclipse Mosquitto
  4. NanoMQ
  5. VerneMQ
- The OpenDDS implementation of DDS (brokerless)

Benchmarking Popular Pub/Sub Systems

We evaluate 6 pub/sub systems using PSMark

- Five MQTT systems (broker-based)
  1. EMQX
  2. Mochi-MQTT
  3. Eclipse Mosquitto
  4. NanoMQ
  5. VerneMQ
- The OpenDDS implementation of DDS (brokerless)

Evaluated using PSMark base metrics:

- Latency, throughput, message loss, CPU/RAM Usage

Benchmarking Popular Pub/Sub Systems

We evaluate 6 pub/sub systems using PSMark

- Five MQTT systems (broker-based)
  1. EMQX
  2. Mochi-MQTT
  3. Eclipse Mosquitto
  4. NanoMQ
  5. VerneMQ
- The OpenDDS implementation of DDS (brokerless)

Evaluated using PSMark base metrics:

- Latency, throughput, message loss, CPU/RAM Usage

Two Quality-of-Service types were mapped to the two protocols

1. “Fire-and-forget” low-QoS settings
2. “Reliable” high-QoS settings

Experimental Setup

Experiments executed in two environments:

1. Single-host commodity server
2. Multi-server Kubernetes cluster*

**Presented results focus on the cluster environment*

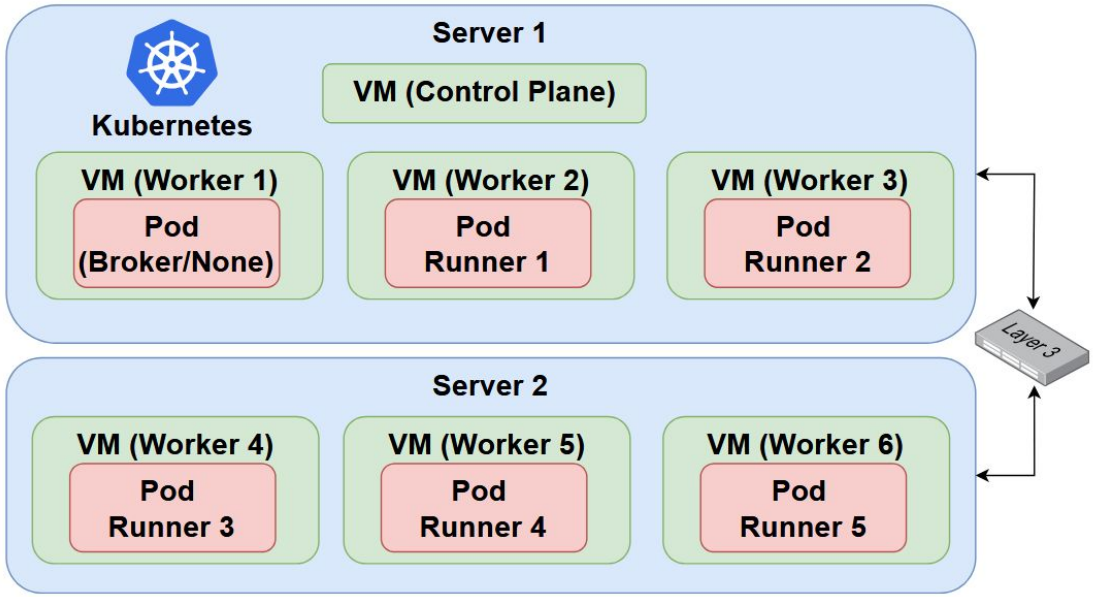
Experimental Setup

Experiments executed in two environments:

1. Single-host commodity server
2. Multi-server Kubernetes cluster*

In the cluster environment:

- 3 VMs per physical server
  - First VM hosting the broker
- Server 1 hosts the control plane



Presented results focus on the cluster environment

Experimental Setup

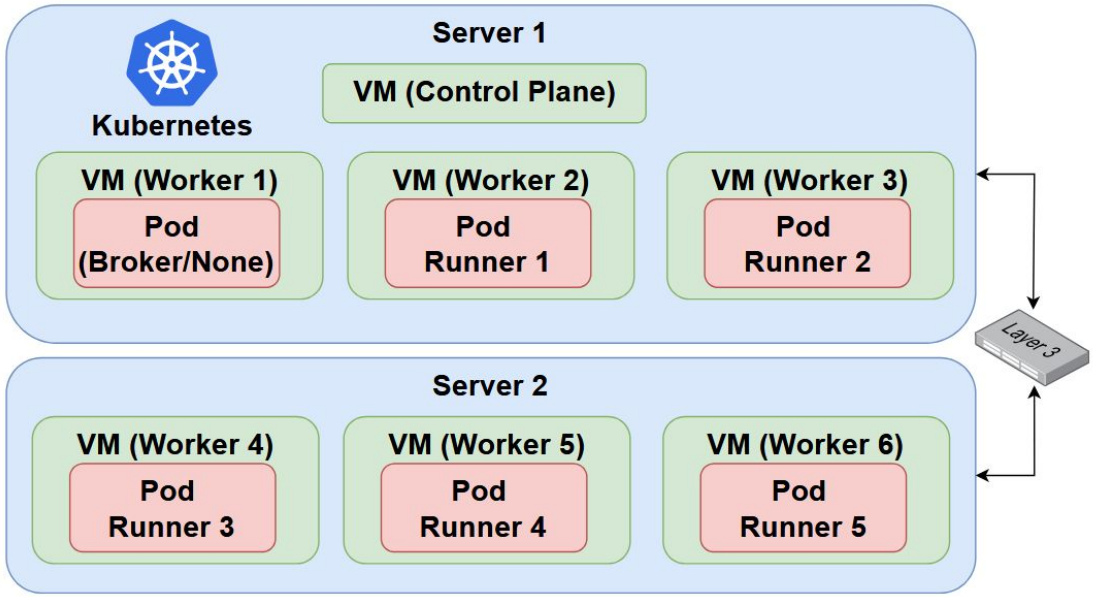
Experiments executed in two environments:

1. Single-host commodity server
2. Multi-server Kubernetes cluster*

Grid'5000

In the cluster environment:

- 3 VMs per physical server
  - First VM hosting the broker
- Server 1 hosts the control plane



For each workload:

- Base scalings run under both high and low QoS settings
- 2x / 10x scalings run under low QoS
- Used default pub/sub system settings

Presented results focus on the cluster environment

Experimental Results

Low-QoS Metrics

	EMQX$_{LQ}$			*Mochi-MQTT*$_{LQ}$			*Mosquitto*$_{LQ}$		
	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)
C	14.93	691	0.0	11.12	690	0.0	11.96	689	0.1
C-2x	34.85	1381	<0.1	27.17	1381	<0.1	31.65	1328	3.9
C-10x	187.59	6913	<0.1	167.29	6578	4.9	83.79	4648	32.8
F	0.89	2725	<0.1	15.38	2826	<0.1	7.75	2830	<0.1
F-2x	0.94	5493	<0.1	1920.27	4219	25.5	3010.44	3311	41.8
HC	1.32	23	0.0	0.93	23	0.0	1.17	23	0.0
HC-2x	1.31	45	0.0	0.82	45	0.0	1.11	45	0.0
HC-10x	10.74	234	0.0	8.49	233	0.0	11.11	234	0.0
HM	1.02	714	0.0	0.90	713	0.0	1.16	715	0.0
HM-2x	0.96	1432	0.0	0.87	1429	0.0	3.31	1431	0.0
HM-10x	0.95	7266	0.0	0.66	7251	0.0	1.14	7256	0.0

	NanoMQ$_{LQ}$			*VerneMQ*$_{LQ}$			*OpenDDS*$_{LQ}$		
	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)
C	48.60	689	<0.1	9.82	686	0.4	11.00	630	8.6
C-2x	106.64	1379	<0.1	19.77	1333	3.5	17.08	1258	8.8
C-10x	4336.30	5474	20.5	86.75	5351	22.5	N/A	N/A	N/A
F	17.40	2831	<0.1	16.98	2830	<0.1	8.56	2802	1.0
F-2x	5042.69	3994	29.2	296.48	3745	34.0	25990.97	5087	10.0
HC	7.05	23	<0.1	1.14	22	0.0	1.05	21	8.1
HC-2x	12.66	45	<0.1	1.07	45	0.0	2.73	41	8.3
HC-10x	68.77	233	0.2	5.91	234	0.0	21.70	212	8.9
HM	1.47	714	0.0	1.08	714	0.0	0.72	708	0.9
HM-2x	1.52	1430	0.0	0.98	1430	0.0	0.77	1415	0.9
HM-10x	3.16	7262	<0.1	0.81	7259	0.0	N/A	N/A	N/A

Low-QoS Latency Percentiles

	EMQX$_{LQ}$			*Mochi-MQTT*$_{LQ}$			*Mosquitto*$_{LQ}$		
	P90	P95	P99	P90	P95	P99	P90	P95	P99
C	42.30	50.28	65.24	31.53	39.34	54.32	34.04	41.27	54.65
F	1.38	1.55	1.93	39.85	46.20	57.79	19.65	25.50	46.75
HC	2.51	3.07	4.41	1.80	2.17	3.25	2.18	2.43	3.27
HM	1.55	1.71	2.01	1.38	1.53	1.80	1.73	1.91	2.32

	NanoMQ$_{LQ}$			*VerneMQ*$_{LQ}$			*OpenDDS*$_{LQ}$		
	P90	P95	P99	P90	P95	P99	P90	P95	P99
C	141.64	169.41	219.48	30.62	38.62	53.68	32.35	46.98	76.06
F	44.39	50.49	57.35	40.02	46.82	61.92	26.19	30.81	38.74
HC	12.18	13.83	16.89	2.14	2.55	4.01	2.16	3.00	5.18
HM	2.61	3.08	4.01	1.66	1.84	2.20	1.16	1.34	1.76

Experimental Results

Low-QoS Metrics

	EMQX $_{LQ}$			*Mochi-MQTT* $_{LQ}$			*Mosquitto* $_{LQ}$		
	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)
C	14.93	691	0.0	11.12	690	0.0	11.96	689	0.1
C-2x	34.85	1381	<0.1	27.17	1381	<0.1	31.65	1328	3.9
C-10x	187.59	6913	<0.1	167.29	6578	4.9	83.79	4648	32.8
F	0.89	2725	<0.1	15.38	2826	<0.1	7.75	2830	<0.1
F-2x	0.94	5493	<0.1	1920.27	4219	25.5	3010.44	3311	41.8
HC	1.32	23	0.0	0.93	23	0.0	1.17	23	0.0
HC-2x	1.31	45	0.0	0.82	45	0.0	1.11	45	0.0
HC-10x	10.74	234	0.0	8.49	233	0.0	11.11	234	0.0
HM	1.02	714	0.0	0.90	713	0.0	1.16	715	0.0
HM-2x	0.96	1432	0.0	0.87	1429	0.0	3.31	1431	0.0
HM-10x	0.95	7266	0.0	0.66	7251	0.0	1.14	7256	0.0

	NanoMQ $_{LQ}$			*VerneMQ* $_{LQ}$			*OpenDDS* $_{LQ}$		
	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)
C	48.60	689	<0.1	9.82	686	0.4	11.00	630	8.6
C-2x	106.64	1379	<0.1	19.77	1333	3.5	17.08	1258	8.8
C-10x	4336.30	5474	20.5	86.75	5351	22.5	N/A	N/A	N/A
F	17.40	2831	<0.1	16.98	2830	<0.1	8.56	2802	1.0
F-2x	5042.69	3994	29.2	296.48	3745	34.0	25990.97	5087	10.0
HC	7.05	23	<0.1	1.14	22	0.0	1.05	21	8.1
HC-2x	12.66	45	<0.1	1.07	45	0.0	2.73	41	8.3
HC-10x	68.77	233	0.2	5.91	234	0.0	21.70	212	8.9
HM	1.47	714	0.0	1.08	714	0.0	0.72	708	0.9
HM-2x	1.52	1430	0.0	0.98	1430	0.0	0.77	1415	0.9
HM-10x	3.16	7262	<0.1	0.81	7259	0.0	N/A	N/A	N/A

EMQX is reliable under low-QoS in taxing scenarios

Factory workloads processed in real-time

Low-QoS Latency Percentiles

	EMQX $_{LQ}$			*Mochi-MQTT* $_{LQ}$			*Mosquitto* $_{LQ}$		
	P90	P95	P99	P90	P95	P99	P90	P95	P99
C	42.30	50.28	65.24	31.53	39.34	54.32	34.04	41.27	54.65
F	1.38	1.55	1.93	39.85	46.20	57.79	19.65	25.50	46.75
HC	2.51	3.07	4.41	1.80	2.17	3.25	2.18	2.43	3.27
HM	1.55	1.71	2.01	1.38	1.53	1.80	1.73	1.91	2.32

	NanoMQ $_{LQ}$			*VerneMQ* $_{LQ}$			*OpenDDS* $_{LQ}$		
	P90	P95	P99	P90	P95	P99	P90	P95	P99
C	141.64	169.41	219.48	30.62	38.62	53.68	32.35	46.98	76.06
F	44.39	50.49	57.35	40.02	46.82	61.92	26.19	30.81	38.74
HC	12.18	13.83	16.89	2.14	2.55	4.01	2.16	3.00	5.18
HM	2.61	3.08	4.01	1.66	1.84	2.20	1.16	1.34	1.76

Experimental Results

Low-QoS Metrics

	$EMQX_{LQ}$			$Mochi\text{-}MQTT_{LQ}$			$Mosquitto_{LQ}$		
	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)
C	14.93	691	0.0	11.12	690	0.0	11.96	689	0.1
C-2x	34.85	1381	<0.1	27.17	1381	<0.1	31.65	1328	3.9
C-10x	187.59	6913	<0.1	167.29	6578	4.9	83.79	4648	32.8
F	0.89	2725	<0.1	15.38	2826	<0.1	7.75	2830	<0.1
F-2x	0.94	5493	<0.1	1920.27	4219	25.5	3010.44	3311	41.8
HC	1.32	23	0.0	0.93	23	0.0	1.17	23	0.0
HC-2x	1.31	45	0.0	0.82	45	0.0	1.11	45	0.0
HC-10x	10.74	234	0.0	8.49	233	0.0	11.11	234	0.0
HM	1.02	714	0.0	0.90	713	0.0	1.16	715	0.0
HM-2x	0.96	1432	0.0	0.87	1429	0.0	3.31	1431	0.0
HM-10x	0.95	7266	0.0	0.66	7251	0.0	1.14	7256	0.0

	$NanoMQ_{LQ}$			$VerneMQ_{LQ}$			$OpenDDS_{LQ}$		
	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)
C	48.60	689	<0.1	9.82	686	0.4	11.00	630	8.6
C-2x	106.64	1379	<0.1	19.77	1333	3.5	17.08	1258	8.8
C-10x	4336.30	5474	20.5	86.75	5351	22.5	N/A	N/A	N/A
F	17.40	2831	<0.1	16.98	2830	<0.1	8.56	2802	1.0
F-2x	5042.69	3994	29.2	296.48	3745	34.0	25990.97	5087	10.0
HC	7.05	23	<0.1	1.14	22	0.0	1.05	21	8.1
HC-2x	12.66	45	<0.1	1.07	45	0.0	2.73	41	8.3
HC-10x	68.77	233	0.2	5.91	234	0.0	21.70	212	8.9
HM	1.47	714	0.0	1.08	714	0.0	0.72	708	0.9
HM-2x	1.52	1430	0.0	0.98	1430	0.0	0.77	1415	0.9
HM-10x	3.16	7262	<0.1	0.81	7259	0.0	N/A	N/A	N/A

EMQX is reliable under low-QoS in taxing scenarios

Factory workloads processed in real-time

Exhibits high latency with large number of unstable clients

Low-QoS Latency Percentiles

	$EMQX_{LQ}$			$Mochi\text{-}MQTT_{LQ}$			$Mosquitto_{LQ}$		
	P90	P95	P99	P90	P95	P99	P90	P95	P99
C	42.30	50.28	65.24	31.53	39.34	54.32	34.04	41.27	54.65
F	1.38	1.55	1.93	39.85	46.20	57.79	19.65	25.50	46.75
HC	2.51	3.07	4.41	1.80	2.17	3.25	2.18	2.43	3.27
HM	1.55	1.71	2.01	1.38	1.53	1.80	1.73	1.91	2.32

	$NanoMQ_{LQ}$			$VerneMQ_{LQ}$			$OpenDDS_{LQ}$		
	P90	P95	P99	P90	P95	P99	P90	P95	P99
C	141.64	169.41	219.48	30.62	38.62	53.68	32.35	46.98	76.06
F	44.39	50.49	57.35	40.02	46.82	61.92	26.19	30.81	38.74
HC	12.18	13.83	16.89	2.14	2.55	4.01	2.16	3.00	5.18
HM	2.61	3.08	4.01	1.66	1.84	2.20	1.16	1.34	1.76

Experimental Results

Low-QoS Metrics

	$EMQX_{LQ}$			$Mochi\text{-}MQTT_{LQ}$			$Mosquitto_{LQ}$		
	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)
C	14.93	691	0.0	11.12	690	0.0	11.96	689	0.1
C-2x	34.85	1381	<0.1	27.17	1381	<0.1	31.65	1328	3.9
C-10x	187.59	6913	<0.1	167.29	6578	4.9	83.79	4648	32.8
F	0.89	2725	<0.1	15.38	2826	<0.1	7.75	2830	<0.1
F-2x	0.94	5493	<0.1	1920.27	4219	25.5	3010.44	3311	41.8
HC	1.32	23	0.0	0.93	23	0.0	1.17	23	0.0
HC-2x	1.31	45	0.0	0.82	45	0.0	1.11	45	0.0
HC-10x	10.74	234	0.0	8.49	233	0.0	11.11	234	0.0
HM	1.02	714	0.0	0.90	713	0.0	1.16	715	0.0
HM-2x	0.96	1432	0.0	0.87	1429	0.0	3.31	1431	0.0
HM-10x	0.95	7266	0.0	0.66	7251	0.0	1.14	7256	0.0

	$NanoMQ_{LQ}$			$VerneMQ_{LQ}$			$OpenDDS_{LQ}$		
	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)
C	48.60	689	<0.1	9.82	686	0.4	11.00	630	8.6
C-2x	106.64	1379	<0.1	19.77	1333	3.5	17.08	1258	8.8
C-10x	4336.30	5474	20.5	86.75	5351	22.5	N/A	N/A	N/A
F	17.40	2831	<0.1	16.98	2830	<0.1	8.56	2802	1.0
F-2x	5042.69	3994	29.2	296.48	3745	34.0	25990.97	5087	10.0
HC	7.05	23	<0.1	1.14	22	0.0	1.05	21	8.1
HC-2x	12.66	45	<0.1	1.07	45	0.0	2.73	41	8.3
HC-10x	68.77	233	0.2	5.91	234	0.0	21.70	212	8.9
HM	1.47	714	0.0	1.08	714	0.0	0.72	708	0.9
HM-2x	1.52	1430	0.0	0.98	1430	0.0	0.77	1415	0.9
HM-10x	3.16	7262	<0.1	0.81	7259	0.0	N/A	N/A	N/A

Mochi-MQTT shows low latency for non-factory workloads

Low-QoS Latency Percentiles

	$EMQX_{LQ}$			$Mochi\text{-}MQTT_{LQ}$			$Mosquitto_{LQ}$		
	P90	P95	P99	P90	P95	P99	P90	P95	P99
C	42.30	50.28	65.24	31.53	39.34	54.32	34.04	41.27	54.65
F	1.38	1.55	1.93	39.85	46.20	57.79	19.65	25.50	46.75
HC	2.51	3.07	4.41	1.80	2.17	3.25	2.18	2.43	3.27
HM	1.55	1.71	2.01	1.38	1.53	1.80	1.73	1.91	2.32

	$NanoMQ_{LQ}$			$VerneMQ_{LQ}$			$OpenDDS_{LQ}$		
	P90	P95	P99	P90	P95	P99	P90	P95	P99
C	141.64	169.41	219.48	30.62	38.62	53.68	32.35	46.98	76.06
F	44.39	50.49	57.35	40.02	46.82	61.92	26.19	30.81	38.74
HC	12.18	13.83	16.89	2.14	2.55	4.01	2.16	3.00	5.18
HM	2.61	3.08	4.01	1.66	1.84	2.20	1.16	1.34	1.76

Experimental Results

Low-QoS Metrics

	$EMQX_{LQ}$			$Mochi\text{-}MQTT_{LQ}$			$Mosquitto_{LQ}$		
	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)
C	14.93	691	0.0	11.12	690	0.0	11.96	689	0.1
C-2x	34.85	1381	<0.1	27.17	1381	<0.1	31.65	1328	3.9
C-10x	187.59	6913	<0.1	167.29	6578	4.9	83.79	4648	32.8
F	0.89	2725	<0.1	15.38	2826	<0.1	7.75	2830	<0.1
F-2x	0.94	5493	<0.1	1920.27	4219	25.5	3010.44	3311	41.8
HC	1.32	23	0.0	0.93	23	0.0	1.17	23	0.0
HC-2x	1.31	45	0.0	0.82	45	0.0	1.11	45	0.0
HC-10x	10.74	234	0.0	8.49	233	0.0	11.11	234	0.0
HM	1.02	714	0.0	0.90	713	0.0	1.16	715	0.0
HM-2x	0.96	1432	0.0	0.87	1429	0.0	3.31	1431	0.0
HM-10x	0.95	7266	0.0	0.66	7251	0.0	1.14	7256	0.0

	$NanoMQ_{LQ}$			$VerneMQ_{LQ}$			$OpenDDS_{LQ}$		
	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)
C	48.60	689	<0.1	9.82	686	0.4	11.00	630	8.6
C-2x	106.64	1379	<0.1	19.77	1333	3.5	17.08	1258	8.8
C-10x	4336.30	5474	20.5	86.75	5351	22.5	N/A	N/A	N/A
F	17.40	2831	<0.1	16.98	2830	<0.1	8.56	2802	1.0
F-2x	5042.69	3994	29.2	296.48	3745	34.0	25990.97	5087	10.0
HC	7.05	23	<0.1	1.14	22	0.0	1.05	21	8.1
HC-2x	12.66	45	<0.1	1.07	45	0.0	2.73	41	8.3
HC-10x	68.77	233	0.2	5.91	234	0.0	21.70	212	8.9
HM	1.47	714	0.0	1.08	714	0.0	0.72	708	0.9
HM-2x	1.52	1430	0.0	0.98	1430	0.0	0.77	1415	0.9
HM-10x	3.16	7262	<0.1	0.81	7259	0.0	N/A	N/A	N/A

OpenDDS shows low reliability under default low-QoS configurations

Low-QoS Latency Percentiles

	$EMQX_{LQ}$			$Mochi\text{-}MQTT_{LQ}$			$Mosquitto_{LQ}$		
	P90	P95	P99	P90	P95	P99	P90	P95	P99
C	42.30	50.28	65.24	31.53	39.34	54.32	34.04	41.27	54.65
F	1.38	1.55	1.93	39.85	46.20	57.79	19.65	25.50	46.75
HC	2.51	3.07	4.41	1.80	2.17	3.25	2.18	2.43	3.27
HM	1.55	1.71	2.01	1.38	1.53	1.80	1.73	1.91	2.32

	$NanoMQ_{LQ}$			$VerneMQ_{LQ}$			$OpenDDS_{LQ}$		
	P90	P95	P99	P90	P95	P99	P90	P95	P99
C	141.64	169.41	219.48	30.62	38.62	53.68	32.35	46.98	76.06
F	44.39	50.49	57.35	40.02	46.82	61.92	26.19	30.81	38.74
HC	12.18	13.83	16.89	2.14	2.55	4.01	2.16	3.00	5.18
HM	2.61	3.08	4.01	1.66	1.84	2.20	1.16	1.34	1.76

Experimental Results

High-QoS Metrics

	EMQX $_{HQ}$			*Mochi-MQTT* $_{HQ}$			*Mosquitto* $_{HQ}$		
	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)
C	38.12	588	14.8	31.85	690	0.0	38.69	566	17.8
F	1.86	2729	<0.1	15.28	2830	0.0	53.66	1039	<0.1
HC	5.87	23	0.0	3.36	23	0.0	7.22	22	0.0
HM	1.47	716	0.0	1.40	714	0.0	2.12	710	0.0

	NanoMQ $_{HQ}$			*VerneMQ* $_{HQ}$			*OpenDDS* $_{HQ}$		
	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)
C	113.78	688	<0.1	65.47	565	18.0	45.00	688	<0.1
F	N/A	N/A	N/A	292.09	2633	6.6	9.22	2808	0.6
HC	15.90	22	<0.1	11.83	23	0.0	18.15	22	0.0
HM	2.01	714	0.0	†2.04	†714	†0.0	1.47	712	0.2

†Results represent 5 successful tests.

Experimental Results

High-QoS Metrics

	EMQX $_{HQ}$			*Mochi-MQTT* $_{HQ}$			*Mosquitto* $_{HQ}$		
	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)
C	38.12	588	14.8	31.85	690	0.0	38.69	566	17.8
F	1.86	2729	<0.1	15.28	2830	0.0	53.66	1039	<0.1
HC	5.87	23	0.0	3.36	23	0.0	7.22	22	0.0
HM	1.47	716	0.0	1.40	714	0.0	2.12	710	0.0

	NanoMQ $_{HQ}$			*VerneMQ* $_{HQ}$			*OpenDDS* $_{HQ}$		
	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)
C	113.78	688	<0.1	65.47	565	18.0	45.00	688	<0.1
F	N/A	N/A	N/A	292.09	2633	6.6	9.22	2808	0.6
HC	15.90	22	<0.1	11.83	23	0.0	18.15	22	0.0
HM	2.01	714	0.0	†2.04	†714	†0.0	1.47	712	0.2

†Results represent 5 successful tests.

Mochi-MQTT did not exhibit any loss under high-QoS

Also shows low latency for most workloads

Experimental Results

High-QoS Metrics

	EMQX $_{HQ}$			*Mochi-MQTT* $_{HQ}$			*Mosquitto* $_{HQ}$		
	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)
C	38.12	588	14.8	31.85	690	0.0	38.69	566	17.8
F	1.86	2729	<0.1	15.28	2830	0.0	53.66	1039	<0.1
HC	5.87	23	0.0	3.36	23	0.0	7.22	22	0.0
HM	1.47	716	0.0	1.40	714	0.0	2.12	710	0.0

	NanoMQ $_{HQ}$			*VerneMQ* $_{HQ}$			*OpenDDS* $_{HQ}$		
	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)
C	113.78	688	<0.1	65.47	565	18.0	45.00	688	<0.1
F	N/A	N/A	N/A	292.09	2633	6.6	9.22	2808	0.6
HC	15.90	22	<0.1	11.83	23	0.0	18.15	22	0.0
HM	2.01	714	0.0	†2.04	†714	†0.0	1.47	712	0.2

†Results represent 5 successful tests.

NanoMQ failed to process high-QoS factory workloads

VerneMQ sometimes failed to process high-QoS home workloads

Experimental Results

High-QoS Metrics

	EMQX$_{HQ}$			*Mochi-MQTT*$_{HQ}$			*Mosquitto*$_{HQ}$		
	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)
C	38.12	588	14.8	31.85	690	0.0	38.69	566	17.8
F	1.86	2729	<0.1	15.28	2830	0.0	53.66	1039	<0.1
HC	5.87	23	0.0	3.36	23	0.0	7.22	22	0.0
HM	1.47	716	0.0	1.40	714	0.0	2.12	710	0.0

	NanoMQ$_{HQ}$			*VerneMQ*$_{HQ}$			*OpenDDS*$_{HQ}$		
	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)	Lat (ms)	Tput (msg/s)	Drop (%)
C	113.78	688	<0.1	65.47	565	18.0	45.00	688	<0.1
F	N/A	N/A	N/A	292.09	2633	6.6	9.22	2808	0.6
HC	15.90	22	<0.1	11.83	23	0.0	18.15	22	0.0
HM	2.01	714	0.0	†2.04	†714	†0.0	1.47	712	0.2

†Results represent 5 successful tests.

OpenDDS exhibits significantly better reliability under high-QoS configurations

Experimental Results

Low-QoS CPU Usage

	$EMQX_{LQ}$	$Mochi_{LQ}$	$Mosquitto_{LQ}$	$NanoMQ_{LQ}$	$VerneMQ_{LQ}$	$OpenDDS_{LQ}$ (Delta)
C	4.13	1.86	1.41	3.80	3.33	2.95
C-2x	7.58	3.07	2.01	7.20	5.35	6.12
C-10x	28.02	14.10	7.60	33.85	25.06	N/A
F	15.25	24.54	12.36	12.92	22.56	3.49
F-2x	20.69	30.47	21.60	25.75	42.03	9.40
HC	0.82	0.51	0.55	0.66	0.50	0.06
HC-2x	1.12	0.44	0.47	0.62	0.63	0.06
HC-10x	1.38	0.91	0.69	1.54	1.05	0.81
HM	6.94	2.49	1.53	3.68	4.00	1.50
HM-2x	10.20	4.60	1.95	5.44	7.10	3.09
HM-10x	25.89	12.09	5.58	12.91	18.63	N/A

Experimental Results

Low-QoS CPU Usage

	$EMQX_{LQ}$	$Mochi_{LQ}$	$Mosquitto_{LQ}$	$NanoMQ_{LQ}$	$VerneMQ_{LQ}$	$OpenDDS_{LQ}$ (Delta)
C	4.13	1.86	1.41	3.80	3.33	2.95
C-2x	7.58	3.07	2.01	7.20	5.35	6.12
C-10x	28.02	14.10	7.60	33.85	25.06	N/A
F	15.25	24.54	12.36	12.92	22.56	3.49
F-2x	20.69	30.47	21.60	25.75	42.03	9.40
HC	0.82	0.51	0.55	0.66	0.50	0.06
HC-2x	1.12	0.44	0.47	0.62	0.63	0.06
HC-10x	1.38	0.91	0.69	1.54	1.05	0.81
HM	6.94	2.49	1.53	3.68	4.00	1.50
HM-2x	10.20	4.60	1.95	5.44	7.10	3.09
HM-10x	25.89	12.09	5.58	12.91	18.63	N/A

Mosquitto consistently uses the least CPU

Experimental Results

Low-QoS CPU Usage

	$EMQX_{LQ}$	$Mochi_{LQ}$	$Mosquitto_{LQ}$	$NanoMQ_{LQ}$	$VerneMQ_{LQ}$	$OpenDDS_{LQ}$ (Delta)
C	4.13	1.86	1.41	3.80	3.33	2.95
C-2x	7.58	3.07	2.01	7.20	5.35	6.12
C-10x	28.02	14.10	7.60	33.85	25.06	N/A
F	15.25	24.54	12.36	12.92	22.56	3.49
F-2x	20.69	30.47	21.60	25.75	42.03	9.40
HC	0.82	0.51	0.55	0.66	0.50	0.06
HC-2x	1.12	0.44	0.47	0.62	0.63	0.06
HC-10x	1.38	0.91	0.69	1.54	1.05	0.81
HM	6.94	2.49	1.53	3.68	4.00	1.50
HM-2x	10.20	4.60	1.95	5.44	7.10	3.09
HM-10x	25.89	12.09	5.58	12.91	18.63	N/A

Mosquitto consistently uses the least CPU

EMQX's CPU usage shows a tradeoff between its reliability and its resource usage

Experimental Results

Low-QoS CPU Usage

	$EMQX_{LQ}$	$Mochi_{LQ}$	$Mosquitto_{LQ}$	$NanoMQ_{LQ}$	$VerneMQ_{LQ}$	$OpenDDS_{LQ}$ (Delta)
C	4.13	1.86	1.41	3.80	3.33	2.95
C-2x	7.58	3.07	2.01	7.20	5.35	6.12
C-10x	28.02	14.10	7.60	33.85	25.06	N/A
F	15.25	24.54	12.36	12.92	22.56	3.49
F-2x	20.69	30.47	21.60	25.75	42.03	9.40
HC	0.82	0.51	0.55	0.66	0.50	0.06
HC-2x	1.12	0.44	0.47	0.62	0.63	0.06
HC-10x	1.38	0.91	0.69	1.54	1.05	0.81
HM	6.94	2.49	1.53	3.68	4.00	1.50
HM-2x	10.20	4.60	1.95	5.44	7.10	3.09
HM-10x	25.89	12.09	5.58	12.91	18.63	N/A

Mosquitto consistently uses the least CPU

EMQX's CPU usage shows a tradeoff between its reliability and its resource usage

OpenDDS has minimal impact on CPU usage for light workloads

Rankings Derived from Experimental Results

PSMark allows straightforward comparison of pub/sub systems

Rankings Derived from Experimental Results

PSMark allows straightforward comparison of pub/sub systems

Top 3 systems per domain

Rankings Derived from Experimental Results

PSMark allows straightforward comparison of pub/sub systems

Top 3 systems per domain

Goal: Minimal Latency at Low-QoS

PSMark-C	PSMark-F	PSMark-HC	PSMark-HM
VerneMQ	*EMQX*	*Mochi-MQTT*	*OpenDDS*
Mochi-MQTT	*Mosquitto*	*VerneMQ*	*Mochi-MQTT*
Mosquitto	*Mochi-MQTT*	*Mosquitto*	*EMQX*

Rankings Derived from Experimental Results

PSMark allows straightforward comparison of pub/sub systems

Top 3 systems per domain

Goal: Minimal Latency at Low-QoS

PSMark-C	PSMark-F	PSMark-HC	PSMark-HM
VerneMQ	*EMQX*	*Mochi-MQTT*	*OpenDDS*
Mochi-MQTT	*Mosquitto*	*VerneMQ*	*Mochi-MQTT*
Mosquitto	*Mochi-MQTT*	*Mosquitto*	*EMQX*

Goal: Maximum Reliability at High-QoS

PSMark-C	PSMark-F	PSMark-HC	PSMark-HM
Mochi-MQTT	*Mochi-MQTT*	*Mochi-MQTT*	*Mochi-MQTT*
OpenDDS	*EMQX*	*EMQX*	*EMQX*
NanoMQ	*Mosquitto*	*Mosquitto*	*NanoMQ*

Overall Experimental Observations

Overall Experimental Observations

- **Workload composition matters more than raw volume**
  - PSMark-F-2x (80 devices; 1,180 msgs/s) stressed most systems
  - PSMark-HM-10x (200 devices; 1,480 msgs/s) did not
  - Capturing real-world device behaviors over only raw message counts is critical

Overall Experimental Observations

- **Workload composition matters more than raw volume**
  - PSMark-F-2x (80 devices; 1,180 msgs/s) stressed most systems
  - PSMark-HM-10x (200 devices; 1,480 msgs/s) did not
  - Capturing real-world device behaviors over only raw message counts is critical

- **QoS–throughput trade-offs are steep at scale**
  - High-QoS settings significantly increased latency and loss
  - Arbitrarily increasing QoS does not guarantee reliability

Key Takeaways

Key Takeaways

- **PSMark provides a distributed pub/sub benchmarking framework for IoT workloads**
  - Measures end-to-end behavior while capturing IoT device heterogeneity
  - Modular support for additional metrics, client interfaces, and protocols

Key Takeaways

- **PSMark provides a distributed pub/sub benchmarking framework for IoT workloads**
  - Measures end-to-end behavior while capturing IoT device heterogeneity
  - Modular support for additional metrics, client interfaces, and protocols
- **Supports IoT researchers and implementers in system evaluation**
  - Compare pub/sub system performance on distributed topologies
  - Evaluate systems across 12 built-in workloads
  - Easily construct custom workloads by composing reusable device definitions

Key Takeaways

- **PSMark provides a distributed pub/sub benchmarking framework for IoT workloads**
  - Measures end-to-end behavior while capturing IoT device heterogeneity
  - Modular support for additional metrics, client interfaces, and protocols
- **Supports IoT researchers and implementers in system evaluation**
  - Compare pub/sub system performance on distributed topologies
  - Evaluate systems across 12 built-in workloads
  - Easily construct custom workloads by composing reusable device definitions
- **Future Work**
  - Control plane for network fault injection
  - Policy-/rule-based messaging to emulate complex configurations
  - Additional protocols (e.g., AMQP, content-based subscription protocols)
  - Additional metrics (e.g., network / client fault tolerance)

IEEE PerCom 2026
Pisa, Italy | March 16 - 20

The 24th IEEE International Conference
on Pervasive Computing and Communications

PSMark is publicly available and ready to be used for IoT benchmarking under built-in or custom workloads!

Code Reviewed

Dataset Reviewed

github.com/DAMSlabUMBC/PSMark

UNIVERSITY OF PATRAS
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

U of Patras is hiring 1 PhD position for a Horizon EU Project starting in October

Topic: *Edge AI*

Horizon Europe